ARCHITECTURE

From RAG to Autonomous Agents

Menlo Ventures

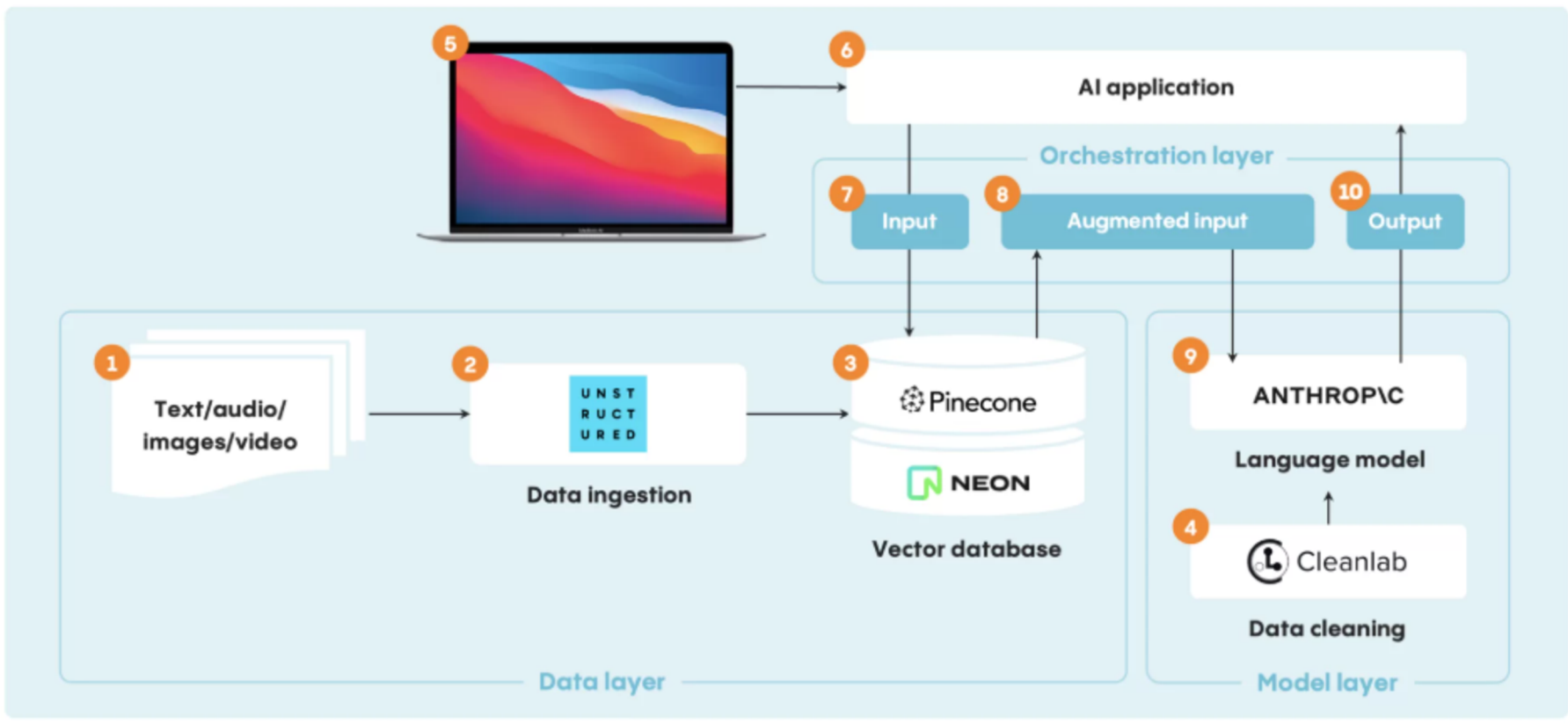
1

Retrieval-Augmented Generation (RAG)

Brief:

RAG is the standard architecture for most modern GenAI applications today. Here's how it works in an enterprise search use case.

- The app loads and transforms unstructured files (e.g. PDFs, slides) from sources like Google Drive and Notion into LLM-readable formats via a preprocessing tool like Unstructured.
- Files are split into smaller text chunks for precise retrieval, embedded as vectors, and stored in a database like Pinecone.
- When a user asks a question (e.g. “Summarize my meeting notes with Company X”), the system retrieves semantically relevant text chunks.
- These chunks are added to a “metaprompt” and fed to the LLM.
- The LLM synthesizes the retrieved context into a concise, bulleted response.



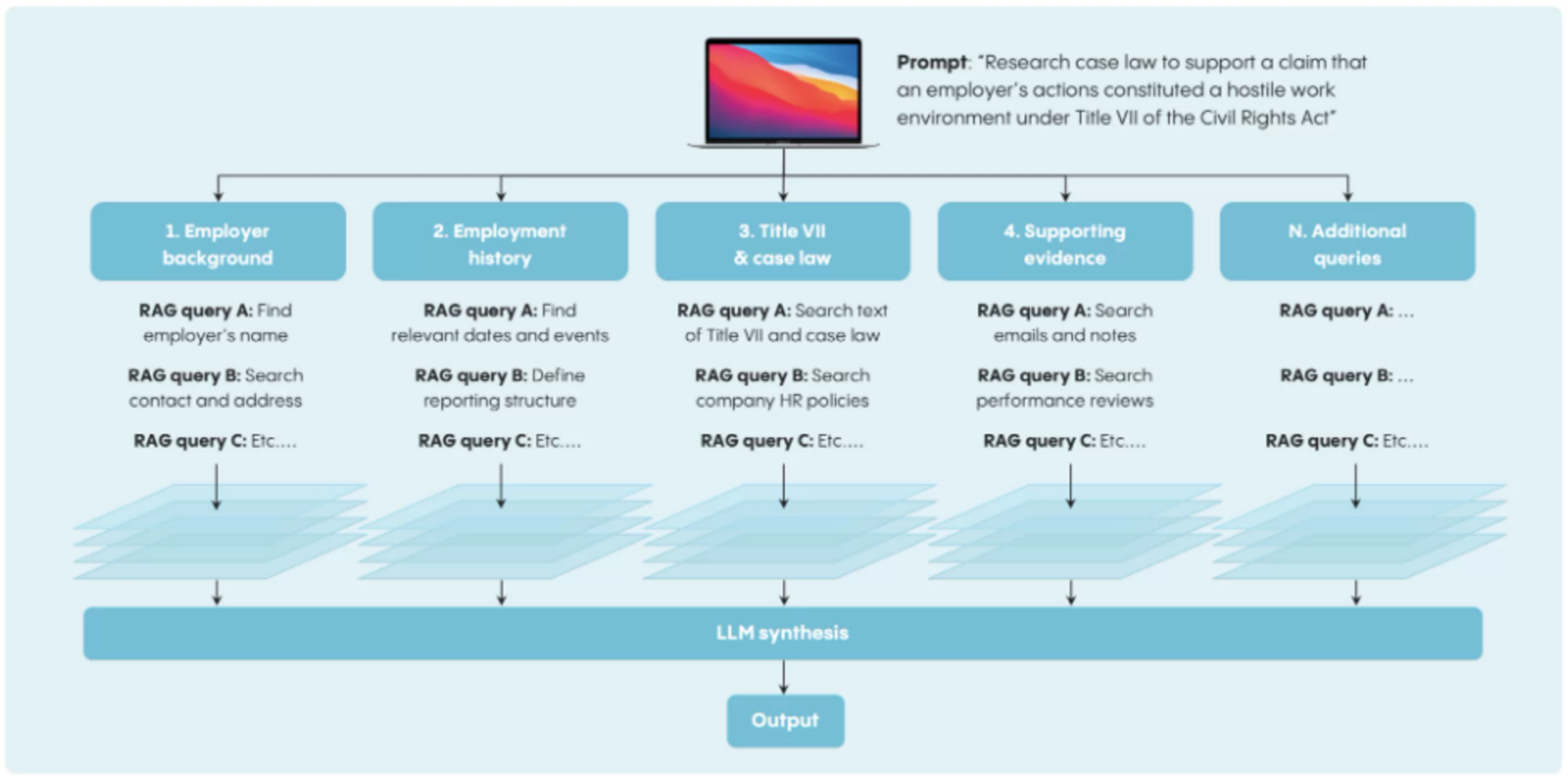
2

RAG + Prompt Chaining

Brief:

The previous example shows a single retrieval step with one LLM call, but production GenAI apps are far complex. Below is an example.

- Apps often use "prompt chains" where outputs from one step feed into the next, and multiple chains run in parallel for different tasks. Results are then combined into a final output.



3

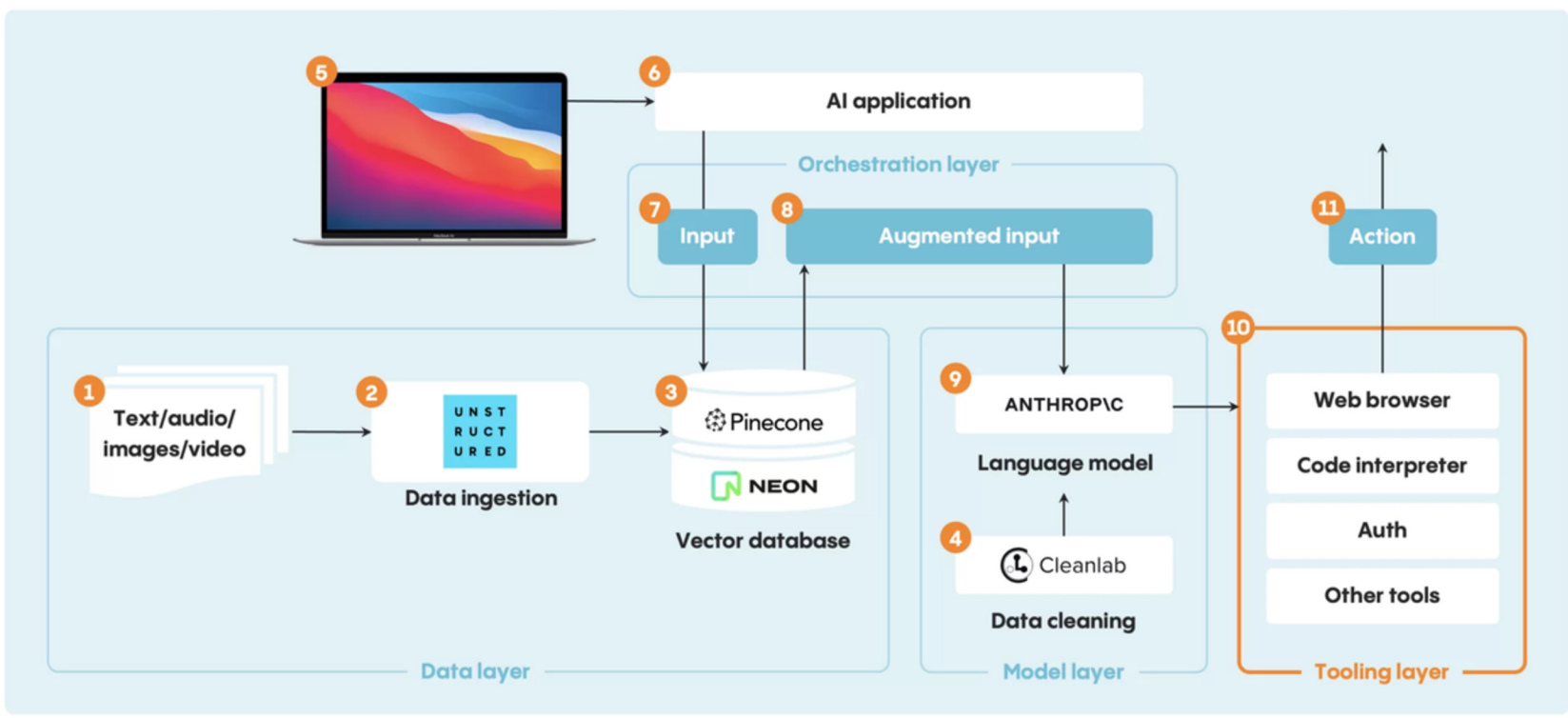
Tool Use

Brief:

Tool use, or function calling, is often seen as the first step from RAG towards agentic behavior, adding a new layer to the GenAI stack.

- These tools, essentially pre-written code components, perform specific actions like web browsing, code interpretation, and authentication. They empower LLMs to navigate the web, interact with external software (e.g. CRMs, ERPs), and run custom code.
- The system presents tools to the LLM, which selects one, prepares inputs as structured JSON, and triggers API executions.
- However, tool use isn't truly "agentic" since control flows are pre-defined by the application. True agents enable LLMs to dynamically write part or all of their own logic.

Reference Architecture: Tool Use



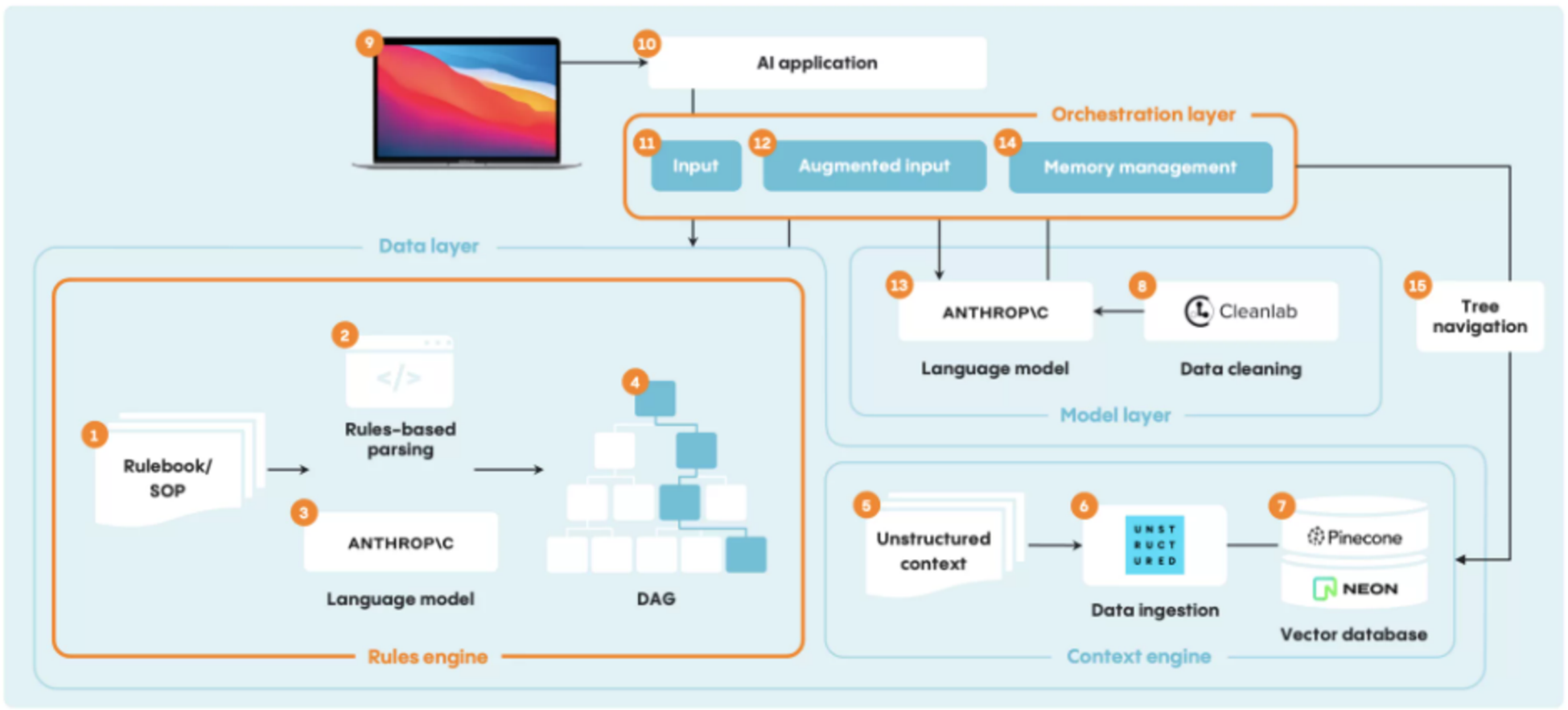
4

Decisioning Agent

Brief:

The first type of agents are decisioning agents, designed to handle complex, multi-step reasoning for business decisions.

- Unlike RAG or tool-use models, they give LLMs limited control over logic, moving away from fully hard-coded steps.
- However, they remain at the lower end of agentic freedom, acting mainly as routers within a set decision tree.



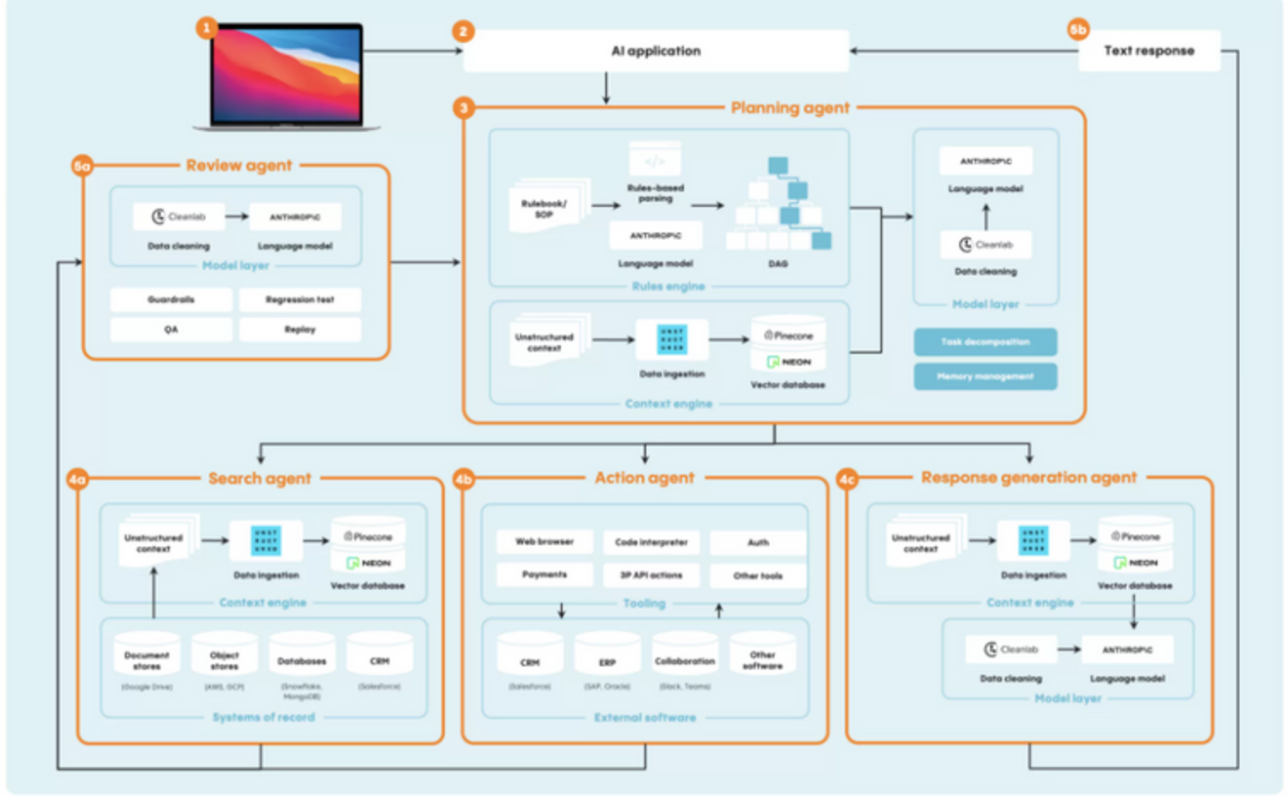
5

Agent on Rails

Brief:

Agents on rails represent the next evolution, given higher-order goals (e.g. “reconcile an invoice” or “help customer troubleshoot a login issue”) and more freedom to choose approaches and tools to achieve goals.

- However, they operate within organizational “rails” (represented as a rulebook or instruction manual written in natural language); given predefined tools enabling set actions in external software systems; and bound by guardrails and other review measures to prevent hallucination. Here’s how it works at runtime:
  - **Planning:** The agent evaluates its current state in the workflow.
  - **Action Selection:** It selects and executes the best action chain (pre-written steps, other agents, or RAG workflows)
  - **Review:** Actions are validated through guardrails before execution.
  - **Reassessment:** The agent updates its state and repeats the process.
- This architecture introduces another order of complexity to previous designs, which can require additional data infrastructure for durable execution; state and memory management for episodic, working, and long-term memory; multi-agent orchestration; and guardrails.



6

General AI Agent

Brief:

The final, still unattainable holy grail of agentic designs is the general AI agent—a for-loop architecture where the LLM's advanced capabilities replace the structured "rails" of previous designs.

- This hypothetical agent would possess dynamic reasoning, planning, and custom code generation abilities, enabling it to perform any action in external systems, not just predefined ones.
- Recent breakthroughs, like BabyAGI and AutoGPT, have advanced this concept. A sophisticated model example is Language Agent Tree Search (LATS), which adapts Monte Carlo Tree Search to help agents explore multiple paths, prioritize high-reward actions, and incorporate feedback.
- Pioneering commercial applications of these frontier architectures include new foundation models like Reflection AI, as well as coding agents like Cognition, Nustom, and OpenDevin/All Hands AI.

Attribution

This document covers key takeaways from Menlo Venture's article, "AI Agents: A New Architecture for Enterprise Automation" (2024. Read in full here: https://menlovc.com/perspective/ai-agents-a-new-architecture-for-enterprise-automation/

Want to level up with Generative AI? Follow  Lewis Walker

Repost this to help your network